Amendments to Specification:

Please replace the paragraph beginning at page 5, line 4 with the following amended paragraph:

Turning now to Fig. 1, illustrative computer architecture for a personal computer 2 for practicing the various embodiments of the invention will be described. The computer architecture shown in Fig. 1 illustrates a conventional personal computer, including a central processing unit 4 ("CPU"), a system memory 6, including a random access memory 8 ("RAM") and a read-only memory ("ROM") 10, and a system bus 12 that couples the memory to the CPU 4. A basic input/output system containing the basic routines that help to transfer information between elements within the computer, such as during startup, is stored in the ROM 10. The personal computer 2 further includes a mass storage device 14 for storing an operating system 16, application programs, such as the application program 305, a schema file 330, and data.

Please replace the paragraph beginning at page 6, line 11 with the following amended paragraph:

As mentioned briefly above, a number of program modules and data files may be stored in the mass storage device 14 and RAM 8 of the personal computer 2, including an operating system 16 suitable for controlling the operation of a networked personal computer, such as the WINDOWS XP operating system from MICROSOFT CORPORATION of Redmond, Washington. The mass storage device 14 and RAM 8 may also store one or more application programs. In particular, the mass storage device 14 and RAM 8 may store an application program 305 for creating and editing an electronic document 310. For instance, the application program 305 may comprise a word processing application program, a spreadsheet application, a contact application, and the like. Application programs for creating and editing other types of electronic documents may also be used with the various embodiments of the present invention.

[[A]] The schema file 330, language settings 26, and a namespace/schema library 400, described below, are also shown.

Please replace the paragraph beginning at page 7, line 3 with the following amended paragraph:

A first object 210 may communicate with a second object 220 to obtain information or functionality from the second object 220 by calling the second object 220 via a message call 230. As is well known to those skilled in the art of object-oriented programming environment, the first object 210 may communicate with the second object 220 via application programming interfaces (API) that allow two disparate software objects 210, 220 to communicate with each other in order to obtain information and functionality from each other. For example, if the first object 210 requires the functionality provided by a method contained in the second object 220, the first object 210 may pass a message call 230 to the second object 220 in which the first object identifies the required method and in which the first object passes any required parameters to the second object required by the second object for operating the identified method. Once the second object 220 receives the call from the first object, the second object executes the called method based on the provided parameters in execution operation 240 and sends a return message 250 containing a value obtained from the executed method back to the first object 210.

Please replace the paragraph beginning at page 8, line 23 with the following amended paragraph:

According to embodiments of the present invention, the text and XML markup entered into the document 310 may be saved according to a variety of different file formats and according to the native programming language of the application 305 with which the document 310 is created. For example, the text and XML markup may be saved according to a word processing application, a spreadsheet application, and the like. Alternatively, the text and XML markup entered into the document 310 may be saved as an XML format whereby the text or data,

any applied XML markup, and any formatting such as font, style, paragraph structure, etc. may be saved as an XML representation. Accordingly, downstream or third party applications capable of understanding data saved as XML may open and consume the text or data thus saved as an XML representation. For a detailed discussion of saving text and XML markup and associated formatting and other attributes of a document 310 as XML, see U.S. Patent Application entitled "Word Processing Document Stored in a Single XML File that may be Manipulated by Applications that Understanding XML," U.S. Serial No. 10/187,060, filed June 28, 2002, which is incorporated herein by reference as if fully set out herein. An exemplary schema in accordance with the present invention is disclosed beginning on page 11 in an application entitled "Mixed Content Flexibility," Serial No. 10/726,077, Docket No. 60001.0275USI[[0]]1, filed December 2, 2003, which is hereby incorporated by reference in its entirety.

Please replace the paragraph beginning at page 9, line 12 with the following amended paragraph:

In order to provide a definitional framework for XML markup elements (tags) applied to text or data, as illustrated in Fig. 3, XML schema files are created which contain information necessary for allowing users and consumers of marked up and stored data to understand the XML tagging definitions designed by the creator of the document. Each schema file also referred to in the art as a Namespace or XSD file preferably includes a listing of all XML elements (tags) that may be applied to a document according to a given schema file. For example, a schema file 330, illustrated in Fig. 3, may be a schema file containing definitions of certain XML elements that may be applied to a document 310 including attributes of XML elements or limitations and/or rules associated with text or data that may be annotated with XML elements according to the schema file. For example, referring to the schema file 330 illustrated in Fig. 3, the schema file is identified by [[the]] a Namespace "intro" 335 the schema file includes a root element of <intro card>.

Please replace the paragraph beginning at page 11, line 22 with the following amended paragraph:

Referring still to Fig. 3, a schema validation functionality module 350 having validation instructions 360 is illustrated for validating XML markup applied to a document 310 against an XML schema file 330 attached to or otherwise associated with the document 310, as described above. As described above, the schema file 330 sets out acceptable XML elements and associated attributes and defines rules for the valid annotation of the document 310 with XML markup from an associated schema file 330. For example, as shown in the schema file 330, two child elements <title> and <body> are defined under the root or parent element <intro card>. Attributes 340, 345 defining the acceptable string length of text associated with the child elements <title> and <body> are also illustrated. As described above, if a user attempts to annotate the document 310 with XML markup from a schema file 330 attached to or associated with the document in violation of the XML markup definitions contained in the schema file 330, an invalidity or error state will be presented. For example, if the user attempts to enter a title string exceeding twenty-five characters, that text entry will violate the maximum character length attribute of the <title> element of the schema file 330. In order to validate XML markup applied to a document 310, against an associated schema file 330, a schema validation module 350 is utilized. As should be understood by those skilled in the art, the schema validation module 350 is a software module including computer executable instructions sufficient for comparing XML markup and associated text entered in to a document 310 against an associated or attached XML schema file 330 as the XML markup and associated text is entered in to the document 310.

Please replace the 5 paragraphs beginning at page 13, line 13 with the following amended paragraphs:

Fig. 4 illustrates a computer-generated screen shot 401 for allowing a user to selectively lock or unlock available style settings for application to a document. As is appreciated by those skilled in the art, a typical software application, such as a word processing application, may have

tens or even hundreds of different style and formatting settings available for application to text and data. A typical style setting might include a style setting called "header 1" in which a particular print size, font, paragraph structure and the like are established. Selection of this style and application to a portion of text or to an entire document will apply the properties of the style setting to the selected text or document. Because each style setting may include numerous properties and because a given software application may include tens or even hundreds of style settings, if each style setting is fully instantiated at the time of document save, file sizes become enormous. Moreover, instantiation of all available style settings may prevent the user of a future or varying version of the software application from utilizing different style settings other than those instantiated by the user upon document save.

Referring to Fig. 4, an exemplary user interface 420 for allowing a user to select style settings 430 that may be used with a particular document or text selection or for allowing the user to select style settings that may not be used for a document or given text selection is described. According to embodiments of the present invention, whether formatting styles are marked as locked for use or locked for non-use, those style settings are not instantiated upon document save, but data representing which style settings are locked and which style settings are unlocked is maintained in a separate data array for future reference by a future application program opening a given document. For a detailed description of locking and unlocking styles and formatting changes for use in association with a computer-generated document, see United States Patent Application Serial No. 10/664,734, filed September 18, 2003, entitled "Method and Apparatus For Restricting The Application Of Formatting To The Contents Of An Electronic Document," applicant matter number 60001.0274US01/304205.1, which is expressly incorporated herein by reference as if fully set out herein.

Fig. 5 illustrates a computer-generated screen shot 500 showing a document formatted according to formatting restrictions 545 and a number of different style settings. As shown in Fig. 5, a document is illustrated whereby four separate style settings 510, 520, 530, and 540 have been applied to text selections in the document.

Fig. 6 illustrates a computer-generated screen shot 600 showing a sample XML file according to embodiments of the present invention. According to embodiments of the present invention, all style settings available for use with a given document are enumerated and data representing each style setting is saved as a latent data object in a separate data array apart from the document. For example, if 156 different style and/or formatting settings are available to the document according to the application program creating the document, a latent count of 156 is set. After the total number of latent style or formatting settings is enumerated, a default state of locked for use or locked for non-use is set for the entire set of enumerated latent style and formatting objects. Any exceptions to the default state are then made. For example, if the user defines that only styles "heading 5, heading 6 and heading 7" may be used for a given document, then all available style and formatting settings will be set to a default of locked for non-use and the style settings "heading 5, heading 6 and heading 7" will be set as exceptions to the default state. According to one embodiment of the invention, the default state may be applied to all style and formatting settings comprising a majority of the enumerated style and formatting settings while the remaining settings will be set as exceptions. That is, if 100 file and formatting settings are available and 60 formatting settings are locked for non-use, then the default state will be set as locked for non-use and the remaining 40 style settings will be designated as exceptions to the default state.

Referring to Fig. 6, an XML representation of a word processing document 610 is illustrated. For representing the latent style and formatting objects saved to a separate data array for referenced by the document 610, an XML tag of <w:latentStyles> is utilized. Within the latent style tag, a default state property is set to "on or off" and a latent style count is enumerated. From the example shown in Fig. 6, a latent style count of 156 indicates that 156 different style or formatting settings are available and that the default state for the 156 style settings is "off" meaning that the default state for each of the 156 possible style and formatting settings is that they may be used with the document 610. Following with the example illustrated in Fig. 6, the user has set styles "heading 5, heading 6 and heading 7" 620 as locked for non-use with the document. Accordingly, by saving the XML structure as illustrated in Fig. 6, a subsequent consuming application that is capable of parsing the XML structure will understand that of the 156 potential style and formatting settings for use with the document, styles "heading

5, heading 6 and heading 7" 620 are exceptions to the default state and may not be used with the document. Advantageously, no XML representation must be made of each and every potential style or formatting setting or of the locked or unlocked state for each and every style or formatting setting which would increase file size and processing time dramatically. Thus, the subsequent XML parsing application needs only to reference the separate data array in which is maintained data representative of each of the potential style and formatting settings.